Solving Hidden-Mode Markov Decision Problems

Samuel Ping-Man Choi **Nevin L. Zhang** **Dit-Yan Yeung**
Department of Computer Science,
Hong Kong University of Science and Technology,
Clear Water Bay, Hong Kong, China
{pmchoi,lzhang,dyyeung}@cs.ust.hk

Abstract

Hidden-Mode Markov decision processes (HM-MDPs) are a novel mathematical framework for a subclass of nonstationary reinforcement learning problems where environment dynamics change over time according to a Markov process. HM-MDPs are a special case of partially observable Markov decision processes (POMDPs), and therefore nonstationary problems of this type can in principle be addressed indirectly via existing POMDP algorithms. However, previous research has shown that such an indirect approach is inefficient compared with a direct HM-MDP approach in terms of the model learning time. In this paper, we investigate how to solve HM-MDP problems efficiently by using a direct approach. We exploit the HM-MDP structure and derive an equation for dynamic programming update. Our equation decomposes the value function into a number of components and as a result, substantially reduces the amount of computations in finding optimal policies. Based on the incremental pruning and point-based improvement techniques, a value iteration algorithm is also implemented. Empirical results show that the HM-MDP approach outperforms the POMDP one several order of magnitude with respect to both space requirement and speed.

1 INTRODUCTION

Hidden-mode Markov decision processes (HM-MDPs) [5, 4] are a novel mathematical framework for a subclass of nonstationary reinforcement learning problems. Unlike traditional nonstationary reinforcement learning in which a slowly-varying environment dynamics is assumed, HM-MDPs are for a specific type of nonstationary problems where environmental changes are restricted to a fixed number of modes. Each mode specifies a Markov decision process (MDP) and shares the same state and action spaces[1], but transition and reward functions may differ according to the mode. In addition, modes are not directly observable and evolve over time according to a Markov process. The notion of modes seems to be applicable to many real-world tasks. For instance, one may use two modes to model bull and bear markets in the stock trading environment. For traffic control problems [6, 11], one may use three modes to model morning rush hours, evening rush hours, and non-rush hours.

HM-MDPs are a generalization of MDPs. In addition to standard MDPs, HM-MDPs allow their model parameters to change probabilistically. This feature is important as many real-world tasks are nonstationary in nature and cannot be represented accurately by a fixed model. Nevertheless, HM-MDPs also add uncertainty in the model parameters and makes the problem, in general, more difficult than the MDP ones.

HM-MDPs are a specialization of POMDPs; they can always be converted into POMDPs with an augmented state space. While POMDPs are superior in terms of representational power, HM-MDPs require fewer parameters, and is a more natural formulation for certain types of nonstationary problems. This simplification has shown significant speedup in model learning [5, 4].

This paper considers how HM-MDP problems can be solved efficiently. Obviously, a straightforward approach is to convert an HM-MDP problem into a POMDP one and then applies existing POMDP algorithms to find a solution. Nevertheless, POMDP algorithms do not exploit the special structures of HM-MDPs. Herein, we derive a Bellman equation and develop a value iteration algorithm specifically for HM-

[1] This condition can always be satisfied by taking the union of the state (or action) spaces of different modes.

MDPs. The resultant algorithm is akin to the POMDP approach in concept but more efficient. In particular, the HM-MDP Bellman equation decomposes the value function into a number of components and as a result, substantially reduces the amount of computations in finding optimal policies. Our empirical results show that the new algorithm significantly outperforms the POMDP value iteration with respect to both space requirement and speed.

2 Background

2.1 HM-MDP FORMULATION

This section gives the formulation of HM-MDPs. Mathematically, an HM-MDP consists of a set of modes $\mathcal{M}$, a set of states $\mathcal{S}$, and a set of actions $\mathcal{A}$. Initially, an agent resides in a mode $m \in \mathcal{M}$ and a state $s \in \mathcal{S}$ with the probabilities of π_m and ψ_s respectively. Unlike in POMDP, states in HM-MDPs are fully observable. In other words, the agent knows exactly which state it is current in. The current mode, on the contrary, is not directly observable and must be inferred through observing the state transitions. Based on the obtained information, the agent selects an action a from A. After executing the action, the agent is subsequently transferred to another state s', according to the probability of $y_m(s,a,s')$ prescribed by the state transition function, and receives a reward $r_m(s,a) \in \Re$. At the same time, the mode changes from m to n with the probability of x_{mn} given by the mode transition function[2]. The goal of the agent is to determine a policy which maximizes the sum of the discounted rewards in the long run.

Since HM-MDPs are special POMDPs, an HM-MDP can always be converted into a POMDP by the transformation shown in Figure 1. Note that $\mathcal{S}', \mathcal{A}', \mathcal{T}', \mathcal{R}', \mathcal{Z}', \mathcal{Q}', \Pi'$ are the state space, action space, state transition function, reward function, observation space, observation function, and prior state probabilities of the resulting POMDP respectively.

$$POMDP = (\mathcal{S}', \mathcal{A}', \mathcal{T}', \mathcal{R}', \mathcal{Z}', \mathcal{Q}', \Pi')$$

$$\mathcal{S}' = \mathcal{M} \times \mathcal{S}, \;\; \mathcal{Z}' = \mathcal{S}, \;\; \mathcal{A}' = \mathcal{A}$$

$$\mathcal{T}' : \{p_{ij}(a) = x_{mn} \cdot y_m(s,a,s') \\ \text{where } i = \langle m,s\rangle \in \mathcal{S}', \; j = \langle n,s'\rangle \in \mathcal{S}'\}$$

$$\mathcal{Q}' : \{q_i(a,s') = \begin{cases} 1 & : \text{ if } s = s' \\ 0 & : \text{ if } s \neq s' \end{cases} \\ \text{where } i = \langle m,s\rangle \in \mathcal{S}', \;\; s' \in \mathcal{Z}'\}$$

$$\mathcal{R}' : \{r_i(a) = r_m(s,a) \mid i = \langle m,s\rangle \in \mathcal{S}'\}$$

$$\Pi' = \{\pi'_i = \pi_m \cdot \psi_s \mid i = \langle m,s\rangle \in \mathcal{S}'\}$$

Figure 1: Reformulating HM-MDP into POMDP

2.2 INDIRECT POMDP APPROACH

Now we briefly review the POMDP approach, as this concept is also essential for solving HM-MDP problems. Among the POMDP algorithms [1], this paper is only concerned with the value iteration method.

Solving a POMDP problem means to prescribe actions (i.e., policy) for every possible situation (i.e., state) in order to optimize the total discounted rewards. Since the states of POMDP are not perceivable, the learning agent must decide its action according to the past observations and actions. Nevertheless, it could be a burden to keep the whole observation and action sequence over a long period of time. Fortunately, this sequence can be summarized by a probability distribution over every possible state without sacrificing the optimal solution. The distribution is represented by a fixed-size real-value vector of the dimension $|\mathcal{S}|$ called *belief state*. Hence, the optimal policy becomes a mapping from the belief state space to the action space.

The optimal policy can be obtained through the optimal value function, which satisfies the following Bellman equation:

$$V(b) = \max_{a \in \mathcal{A}} (r(b,a) + \gamma \sum_{z \in \mathcal{Z}} \mathrm{P}(z|b,a) V(b_z^a)) \qquad (1)$$

where $V(\cdot)$ is the value function; b denotes the belief state; γ denotes the discount factor; a is an action in the action space $\mathcal{A}$; z is an observation in the observation space $\mathcal{Z}$; $r(b,a)$ is the immediate reward obtained by taking the action a at the belief state b; b_z^a is the next belief state given the action a and observation z. By using the above equation, the value iteration method iteratively improves the value function until the convergence is reached. Each iteration is known as a *dynamic-programming update*.

Exact POMDP algorithms compute the optimal value function for all possible belief states. Since there are infinite number of belief states, it is infeasible to represent the value function in table form. Fortunately, the value function is *piecewise linear and convex* (PWLC), and thus can be represented by a set of vectors [12].

[2]Herein, we only consider the case that the mode transitions are independent of the chosen actions. In the general case, the mode transitions can also be a function of actions.

It turns out that the dynamic-programming update involves how to generate a new set of vectors based on the old set. This vector set grows exponentially in size but fortunately, many vectors in the set are redundant. Incremental pruning [2] is an efficient technique that prunes the vector set incrementally so as to minimize the number of operations required. Together with the point-based improvement technique [13], incremental pruning is currently known as the most efficient algorithm for solving POMDP problems.

3 SOLVING HM-MDPS

While developing independently, our work happens to be very similar to the one proposed by Hauskrecht and Fraser [7], where they briefly described how the decomposition method can be used for dynamic belief networks with both partially and fully observable states. The idea is then applied to the diagnosis of the ischemic heart disease along with others such as approximation method. However, they do not address in depth how the decomposition method alone can save the computation and the space requirements in finding the optimal solution. In this paper, we elaborate their idea in the context of HM-MDPs and evaluate the approach analytically and empirically. We also demonstrate how the modified Bellman equation can be easily extended to existing value iteration algorithms.

Now let us examine why the POMDP approach is not an ideal choice for solving HM-MDPs. Suppose an HM-MDP and its equivalent POMDP are given. Recall that $\mathcal{M}$ and $\mathcal{S}$ are respectively the sets of modes and states in HM-MDPs. According to Figure 1, a belief state b of the equivalent POMDP is a probability distribution of the current state over the set $\mathcal{M} \times \mathcal{S}$. Such a belief state can have up to $|\mathcal{M}| \cdot |\mathcal{S}|$ non-zero entries so long as $\sum_{\langle m,s\rangle \in \mathcal{M}\times\mathcal{S}} b(\langle m,s\rangle) = 1$. This representation is in fact more general than necessary, because an HM-MDP only has uncertainty in modes but not states. In other words, an HM-MDP belief state should only contain at most $|\mathcal{M}|$ number of non-zero entries. Lemma 1 proves this statement.

Lemma 1 *A valid belief state of an HM-MDP-equivalent POMDP has at most $|\mathcal{M}|$ non-zero entries.*

Proof: Recall that the belief state updating equation for POMDP is the following:

$$b_z^a(s') = \frac{\mathrm{P}(z|s',a)\sum_{s\in\mathcal{S}} \mathrm{P}(s'|s,a)b(s)}{\mathrm{P}(z|b,a)}, \qquad (2)$$

given a belief state b, an action a, and the resulting observation z.

Due to the HM-MDP to POMDP transformation, the hidden states of an HM-MDP-equivalent POMDP can always be written as $\langle m,s\rangle$. By substituting $\langle m,s\rangle$ for s, Equation 2 is rewritten as:

$$b_z^a(\langle m,s\rangle') = \frac{\mathrm{P}(z|\langle m,s\rangle',a)\sum_{\langle m,s\rangle} \mathrm{P}(\langle m,s\rangle'|\langle m,s\rangle,a)b(\langle m,s\rangle)}{\mathrm{P}(z|b,a)} \qquad (3)$$

Since $\mathrm{P}(z|\langle m,s\rangle',a)$ equals 1 if $z=s$ and 0 otherwise, there are exactly $|\mathcal{M}|$ number of $\mathrm{P}(z|\langle m,s\rangle,a)$ giving the value of 1. It follows that b_z^a has at most $|\mathcal{M}|$ non-zero entries. □

Lemma 1 implies that not every belief state in the equivalent POMDP is relevant to the original problem. Many of them simply never occur. From this point of view, the indirect approach unnecessarily enlarges the belief state space and complicates the problem. A more compact representation is thus needed to eliminate the redundant belief states. Exploiting the fact that states in HM-MDPs are fully observable, we introduce a representation that satisfies the constraint $\sum_{m\in\mathcal{M}} b(\langle m,s\rangle) = 1, \ \forall\, s \in \mathcal{S}$.

3.1 BELIEF STATE FOR HM-MDPS

The new belief state for HM-MDP contains two components: the *current state* and the *belief mode*. The current state, as usual, is a discrete variable that keeps the state observed at the current time instance. This is the belief mode that keeps the probability distribution of the current mode over every possible mode. A belief mode is denoted as μ; the set of all possible belief modes is referred to as the *belief mode space* and denoted by $\mathcal{B}_m$.

Lemma 2 *The belief mode plus the current state is sufficient for representing any valid belief state in an HM-MDP-equivalent POMDP.*

Proof: A valid belief state of an HM-MDP-equivalent POMDP must satisfy the belief state updating rule (Equation 3). Lemma 1 shows that the non-zero entries are corresponding to different modes but the same observable state. There is no information loss by keeping only the mode distribution and the current state, because a belief mode can always be converted back into its original belief state. Hence, the lemma follows. □

Lemma 2 implies that the belief mode and the current state is sufficient statistics for HM-MDPs. In other words, the optimal solution can be obtained based on these two pieces of information without the need of referring to the past state transitions.

3.2 UPDATING BELIEF MODES

Let s and μ be the current state and the current belief mode respectively. Suppose an action a is executed and afterwards, the environment moves to state s'. According to the Bayes rule and the basic probability theory, the next belief mode μ' is then given by

$$\mu'(m') = \frac{\sum_m \mathrm{P}(m'|m)\mathrm{P}(s'|s,m,a)\mu(m)}{\sum_{m',m} \mathrm{P}(m'|m)\mathrm{P}(s'|s,m,a)\mu(m)} \quad (4)$$

This equation signifies the dependency of μ' on the action a and state s'; the next belief mode μ' is also denoted as $\mu^a_{s'}$. The numerator computes the likelihood of the next mode, and the denominator is the normalization constant. This constant is sometimes written as $\mathrm{P}(s'|s,\mu,a)$, and can be interpreted as the probability of reaching state s' when executing action a in state s and belief mode μ.

3.3 OPTIMAL POLICIES FOR HM-MDPS

At each point in time, the agent chooses an action based on the current state and the current belief mode. An *HM-MDP policy* prescribes an action for each pair of state and belief mode. In other words, the policy is a mapping from $\mathcal{S} \times \mathcal{B}_m$ to $\mathcal{A}$.

The value function V^π of a policy π is a real-valued function over $\mathcal{S} \times \mathcal{B}_m$. For any state s and any belief mode μ, $V^\pi(s,\mu)$ is the expected discounted long term rewards the agent obtains if it acts according to policy π starting from state s and belief mode μ. Formally, it is given by

$$V^\pi(s,\mu) = E_{s,\mu}[\sum_{t=0}^{\infty} \gamma^t r_t],$$

where r_t is the reward obtained at time t and γ is the discount factor in the range of $[0, 1)$.

Following the proofs in POMDPs and MDPs, it could be shown that there exists a policy π^* such that

$$V^{\pi^*}(s,\mu) \geq V^\pi(s,\mu)$$

for any state s, any belief mode μ and any other policy π. Such a policy is called an *optimal policy*, and its associated value function is called the *optimal value function*, denoted by V^*. For any positive number ϵ, a policy π is *ϵ-optimal* if

$$V^\pi(s,\mu) + \epsilon \geq V^*(s,\mu),$$

for any state s and any belief mode μ.

3.4 VALUE ITERATION

Value iteration is a standard algorithm for finding ϵ-optimal policies for both MDPs and POMDPs. It could also be easily adapted to HM-MDPs. Now the new HM-MDP equation can be derived from the POMDP by using the belief mode μ and the current state s. By substituting the belief mode and the current state for the belief state, Equation 1 is rewritten into the following:

$$V(\mu,s) = \max_{a\in\mathcal{A}}(r(\mu,s,a) + \gamma \sum_{s'\in\mathcal{S}} \mathrm{P}(s'|\mu,s,a)V(\mu',s'))$$

where μ', s' denotes the next belief mode given the action a and the resultant state s'; the reward $r(\mu,s,a)$ can be computed by $\sum_{m\in\mathcal{M}} r_m(s,a)\cdot\mu(m)$. Since s is a discrete variable, one can view the value function V as $|\mathcal{S}|$ number of value functions of the belief mode vectors. Now the above equation becomes:

$$V_s(\mu) = \max_{a\in\mathcal{A}}(\sum_{m\in\mathcal{M}} r_m(s,a)\mu(m) + \gamma \sum_{s'\in\mathcal{S}} \mathrm{P}(s'|\mu,s,a)V_{s'}(\mu')) \quad (5)$$

Note that the decomposed value functions are still piecewise linear and convex (PWLC), since a subset of PWLC function is also PWLC. One can now use Equation 5 for the value iteration to compute the optimal value function for each observable state in $\mathcal{S}$.

Define an operator T that takes a value function V and returns another value function TV as follows:

$$TV_s(\mu) = \max_a[r(s,\mu,a) + \gamma \sum_{s'} \mathrm{P}(s'|s,\mu,a)V_{s'}(\mu')]. \quad (6)$$

where $r(s,\mu,a) = \sum_m r_m(s,a)\mu(m)$ is the expected immediate reward for taking action a in state s and belief mode μ. For a given value function V, a policy π is said to be *V-improving* if

$$\pi(s,\mu) = \arg\max_a[r(s,\mu,a) + \gamma \sum_{s'} \mathrm{P}(s'|s,\mu,a)V_{s'}(\mu')] \quad (7)$$

for all belief modes μ.

3.5 BELLMAN RESIDUAL FOR HM-MDPS

Value iteration is an algorithm for finding ϵ-optimal policies. It starts with an initial value function V^0 and iterates using the following formula:

$$V^n = TV^{n-1}.$$

It is known (e.g. [10, Theorem 6.9]) that V^n converges to V^* as n goes to infinity. Value iteration terminates when the *Bellman residual* $\max_b |V^n(\mu) - V^{n-1}(\mu)|$

falls below $\epsilon(1-\lambda)/2\lambda$. When it does, a V^n-improving policy is ϵ-optimal. As the HM-MDP value function contains a number of components, the Bellman residual for HM-MDP becomes:

$$\max_{\mu\in\mathcal{B},s\in\mathcal{S}} |V_s^n(\mu) - V_s^{n-1}(\mu)|$$

Since there are infinitely many possible belief states, value iteration cannot be carried explicitly. Fortunately, it can be carried out implicitly. In the next section, we discuss how the implicit value iteration can be achieved by manipulating a set of vectors.

4 IMPLICIT VALUE ITERATION

4.1 TECHNICAL AND NOTATIONAL CONSIDERATIONS

We first introduce several technical concepts and notations. Following the convention of POMDP, we call functions over the mode space *vectors*. Lower case Greek letters α and β are used to refer to vectors whereas script letters $\mathcal{V}$ and $\mathcal{U}$, sometimes with subscripts, are used to refer to sets of vectors. In contrast, the upper letters V and U always refer to value functions, i.e., functions over $\mathcal{S} \times \mathcal{B}_m$. Note that a belief mode is a function over the mode space and hence can be viewed as a vector.

Suppose $\mathcal{W}$ and $\mathcal{X}$ are two sets of vectors over the mode space. The *cross sum* of $\mathcal{W}$ and $\mathcal{X}$ is a new set of vectors given by

$$\mathcal{W}\oplus\mathcal{X} = \{\alpha+\beta|\alpha\in\mathcal{W}, \beta\in\mathcal{X}\}.$$

It is evident that the cross sum operation is commutative and associative. Hence one can talk about the cross sum of more than two sets of vectors. For any constant γ, $\gamma\mathcal{W} = \{\gamma\alpha|\alpha \in \mathcal{W}\}$.

A subset $\mathcal{W}'$ of $\mathcal{W}$ is a *covering* of $\mathcal{W}$ if for any belief mode μ, there exists $\alpha'\in\mathcal{W}'$ such that

$$\alpha' \cdot \mu \geq \alpha \cdot b$$

for all $\alpha\in\mathcal{W}$. Here $\alpha'\cdot\mu$ and $\alpha\cdot\mu$ are the inner products of α' and α with μ. A covering of $\mathcal{W}$ is *parsimonious* if none of its proper subsets are coverings of $\mathcal{W}$. If $\mathcal{W}$ is a parsimonious covering of itself, we say that $\mathcal{W}$ is *parsimonious*.

The *witness region* $R(\alpha, \mathcal{W})$ of a vector $\alpha\in\mathcal{W}$ with respect to $\mathcal{W}$ is subsets of $\mathcal{B}_m$ respectively given by

$$R(\alpha, \mathcal{W}) = \{\mu\in\mathcal{B}_m|\alpha \cdot \mu > \alpha' \cdot \mu \quad \forall\alpha'\in\mathcal{W}\backslash\{\alpha\}\},$$

It can be proved [9] that $\mathcal{W}$ has a unique parsimonious covering and is given by

$$PC(\mathcal{W}) = \{\alpha|R(\alpha, \mathcal{W}) \neq \emptyset\}.$$

A point in $R(\alpha, \mathcal{W})$ is called a *witness point* for α because it testifies to the fact that α is in the parsimonious covering $PC(\mathcal{W})$.

4.2 FINITE REPRESENTATION OF VALUE FUNCTIONS

Consider a value function $V(s, \mu)$ and a collection of sets of vectors $\{\mathcal{V}_s|s \in \mathcal{S}\}$. We say that the sets *represent* the value function if:

$$V(s, \mu) = \max_{\alpha\in\mathcal{V}_s} \alpha \cdot \mu, \quad \forall\mu \in \mathcal{B}_m,$$

When this is the case, we say that the value function is *representable* by a finite number of vectors. Note that if $\{\mathcal{V}_s|s \in \mathcal{S}\}$ represents a value function, then so does $\{PC(\mathcal{V}_s)|s \in \mathcal{S}\}$. This representation is *parsimonious* in the obvious sense.

Unless otherwise specified, from now we assume that value iteration begins with the 0 value function, i.e., the value function that is 0 everywhere. This value function is obviously representable by a finite number of vectors. Together with the following lemma, this implies all value functions produced during value iterations are representable by a finite number of vectors.

Lemma 3 *Suppose a value function V is represented by a collection $\{\mathcal{V}_{s'}|s' \in \mathcal{S}\}$ of sets of vectors. Then TV is represented by $\{\mathcal{U}_s|s \in \mathcal{S}\}$, where*

$$\mathcal{U}_s = \cup_a[\{r(s, m, a)\} \oplus \gamma(\oplus_{s'}\mathcal{V}_{s,a,s'})],$$

and $\mathcal{V}_{s,a,s'}$ is a set of vectors given by

$$\mathcal{V}_{s,a,s'} = \{\beta \mid \exists\, \alpha \in \mathcal{V}_{s'} \ s.t.\ \forall m \in \mathcal{M},$$
$$\beta(m) = \sum_{m'} \alpha(m')P(m'|m)P(s'|s, m, a)\ \}$$

This lemma can be proved in a similar way to the corresponding fact for POMDP (see [14, page 210]).

5 DYNAMIC-PROGRAMMING UPDATE FOR HM-MDPS

Dynamic-programming update (DPU) refers to the process of computing a parsimonious representation of TV from a parsimonious representation $\{\mathcal{V}_s|s \in \mathcal{S}\}$ of V. The naive approach to this problem is to first construct the sets $\mathcal{U}_s$ defined above and then compute their parsimonious covering. This approach is inefficient since the size of $\mathcal{U}_s$ is exponential in $|\mathcal{S}|$ and $|\mathcal{A}|$.

5.1 INCREMENTAL PRUNING

A more efficient method called incremental pruning (IP) is employed here. The basic idea of IP is to interleave cross sums and pruning of extraneous vectors, so as to reduces the required computations for each iteration. There exist some variants of the algorithm [2], but here we consider only the basic version. The outline of the HM-MDP IP algorithm is given below:

Procedure DP-UPDATE($\{\mathcal{V}_{s'} | s' \in \mathcal{S}\}$)
1. For each s,
2. For each a,
3. $\mathcal{V}_{s,a} \leftarrow \mathtt{IP}(\{\mathcal{V}_{s,a,s'} | s' \in \mathcal{S}\})$.
4. $\mathcal{U}_s \leftarrow PC(\cup_a[\{r_m(s,a)\} \bigoplus \mathcal{V}_{s,a}])$
5. return $\{\mathcal{U}_s | s \in \mathcal{S}\}$.

Procedure IP($\{\mathcal{W}_i | i = 1, \ldots, m\}$)
1. $\mathcal{Y} \leftarrow \mathcal{W}_1$.
2. **For** $i = 2$ to m,
3. $\mathcal{Y} \leftarrow PC(\mathcal{Y} \oplus \mathcal{W}_i)$.
3. Return $\mathcal{Y}$.

The function DP-UPDATE takes from the input an array of vector sets, performs a dynamic-programming update according to the HM-MDP Bellman equation, and returns the updated vector sets. There is no difference between the HM-MDP and the POMDP versions in the IP procedure. The HM-MDP incremental pruning algorithm differs from the original one mainly in the dynamic-programming update routine (DP-UPDATE). The major difference would be that a collection of (rather than a single) value functions are maintained and indexed by states.

In Equation 1, the summation of the value function for all observations is very costly. It generates new vectors exponential to the number of observations (i.e., $|\mathcal{V}|^{|\mathcal{Z}|}$). Comparing with the POMDP equation, the HM-MDP one is more efficient since $\mathcal{V}_{s'}$ is a subset of $\mathcal{V}$ and therefore contains much fewer vectors in general.

5.2 POINT-BASED IMPROVEMENT TECHNIQUE

The point-based improvement (PBI) technique [13] is an efficient method for speeding up the value iteration process. The idea behind is similar to that of the modified policy iteration method. In particular, PBI approximates the new value function by applying the backup operator on each vectors in the old value function. This procedure is repeated until the largest vector improvement falls below a threshold. PBI can be incorporated easily into any value-iteration method and empirical results show that it substantially reduces the required number of iterations.

For HM-MDPs, one should note that PBI should be applied to every $\mathcal{V}_s$ for each iteration instead of repeatedly on one single function. The complete HM-MDP PBI algorithm is depicted as follows.

Procedure PBI($\{\mathcal{V}_s | s \in \mathcal{S}\}$):
1. $\mathcal{U}_s \leftarrow \mathcal{V}_s \;\; \forall\, s \in \mathcal{S}$
2. **do** {
3. $\mathcal{U}' \leftarrow \emptyset, \mathcal{W} \leftarrow \emptyset$.
4. **For** $s \in \mathcal{S}$,
5. **For** $\alpha \in \mathcal{U}_s,\ s' \in \mathcal{S}$,
6. $\alpha' \leftarrow$ BACKUP(pt(α), act(α), $\mathcal{U}_{s'} \cup \mathcal{W}$)
7. **if** $\alpha \cdot$ pt(α) $>$ $\alpha' \cdot$ pt(α)
8, $\alpha' \leftarrow \alpha$
9. **else**
10. $\mathcal{W} \leftarrow \mathcal{W} \cup \{\alpha'\}$
11. pt(α') $\leftarrow$ pt(α)
12. act(α') $\leftarrow$ act(α)
13. $\mathcal{U}_s \leftarrow \mathcal{U}_s \cup \{\alpha'\}$
14. } **while** stop($\{\mathcal{U}_s\}, \{\mathcal{U}'_s\} | s \in \mathcal{S}$) = false
15. **return** $\{\mathrm{PC}(\mathcal{U}'_s \cup \mathcal{V}_s) | s \in \mathcal{S}\}$

PBI takes as input a collection of vector sets and improves them iteratively. The HM-MDP version of PBI has two additional loops (line 4 and 5) for the current state s and the next state s'. Line 6 constructs a new vector based on the backup operator. Note that only a subset of value function (i.e., $\mathcal{U}_{s'}$) is needed. Line 7 to line 12 are the same as the original version, and pt(α) and act(α) are respectively the witness point and the action associated with the vector α. Line 14 determines if the PBI process should be repeated. Line 15 computes and returns the parsimonious representation of the resultant vector sets based on an efficient method described in [13].

The termination condition relying on a single improvement is sometimes problematic due to the possible precision error in solving the linear programs. We therefore used a slightly different scheme [8] in our implementation. In particular, we determine the termination condition of the recursive improvement call by examining the average vector improvement. This scheme is found more robust empirically for many test cases.

6 EMPIRICAL RESULTS

Our experiments were conducted on a number of simplified real-world domains. Table 1 shows the summary of the tasks. Among them, the first three problems are randomly generated, whereas the fourth and the last two problems are respectively described in [4] and [3]. Note that some of these problems are fairly large in the context of POMDPs, since the number

Table 1: HM-MDP problems

Problem	Modes	States	Actions
Random1	2	2	2
Random2	2	3	3
Random3	2	4	3
Traffic Light	2	8	2
Sailboat	4	16	2
Elevator	3	32	3

Table 2: Empirical results on solving HM-MDP problems using POMDP incremental pruning and point-based improvement

Problem	POMDP Approach		
	Time	Vectors	Epochs
Random1	745	1672	10
Random2	446299	28272	8
Random3	>604800	-	-
Traffic Light	>604800	-	-
Sailboat	>604800	-	-
Elevator	>604800	-	-

of hidden states in the corresponding POMDP problems is equal to the number of observable states multiplied by the number of hidden modes. In the elevator problem, for example, an equivalent POMDP problem would contain 96 hidden states and 32 observations.

We implemented the HM-MDP IP method together with the PBI technique, and ran the program on a SPARC Ultra 2 machine with a time limit of CPU 604800 seconds (7 days) for each task. In the experiments, a discount factor 0.95 and a 0.01-optimal solution were considered. The PBI termination parameter was set at 0.01 of the Bellman residual. The CPU time in seconds (Time), the number of vectors in the value function (Vectors) and the number of iterations (Epochs) are reported in Table 2 and 3.

Table 3: Empirical results on solving HM-MDP problems using HM-MDP incremental pruning and point-based improvement

Problem	HM-MDP Approach		
	Time	Vectors	Epochs
Random1	27	174	10
Random2	41	235	8
Random3	28	163	6
Traffic Light	48	446	9
Sailboat	1815	1342	10
Elevator	725	3154	8

The HM-MDP approach outperforms the POMDP one significantly with respect to both time and space. For the first and second problems, the HM-MDP approach is about 28 times and 10884 times faster than the POMDP approach respectively. For the rest of the problems, the POMDP approach simply cannot complete the tasks within the specified time limit.

Figure 2 and 3 show the actual performance of both algorithms during each iteration. The y-axis is the estimated policy quality based on the Bellman residual while the x-axis is the CPU running time in seconds. Note that both x and y axes are in logarithmic scale. Each data point plotted in the graph indicates one iteration of the run. The figures reveal that the HM-MDP approach performs consistently better than the POMDP one, and the performance gain increases with the number of iterations.

For the space requirement, the number of vectors generated by the HM-MDP approach is also much smaller than the POMDP one — only about 10% and 0.8% of its counterpart in size for the first two problems. It is worth mentioning that the length of vectors in HM-MDP (i.e., $|\mathcal{M}|$) is also much less than that in POMDP (i.e., $|\mathcal{M}| \cdot |\mathcal{S}|$). This means that the actual saving in space is even greater in practice.

From the first two experiments we conducted, the number of iterations required for HM-MDP and POMDP problems are the same. This suggests that PBI for POMDP problems works as effective as for the HM-MDP problems in terms of reducing the number of iterations. However, due to the relatively larger value function, each PBI call typically requires more computational time for the POMDP than the HM-MDP.

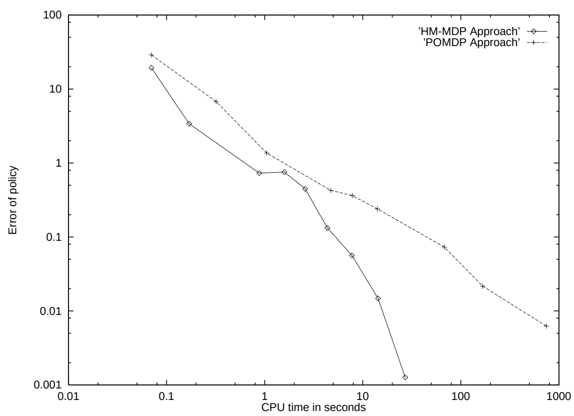

Figure 2: The policy quality over time for the Random1 problem.

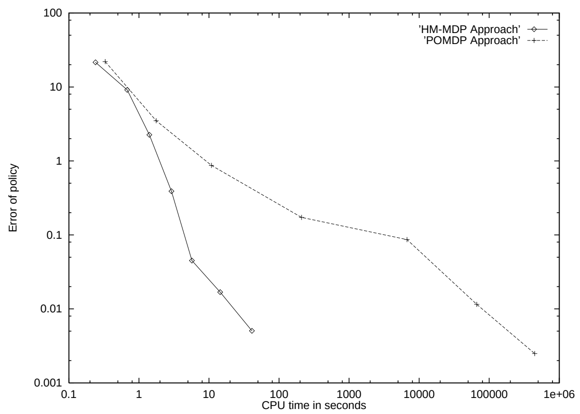

Figure 3: The policy quality over time for the Random2 problem.

7 CONCLUSION

We have derived a variant of Bellman equation for solving HM-MDPs. Since the equation is similar to that of POMDP, existing POMDP algorithms, such as IP and PBI, can easily be adopted. In addition, solving HM-MDPs can directly benefit from any future advances in POMDP techniques.

Empirical results verify that the direct HM-MDP approach is more efficient than the indirect POMDP approach. There are two main reasons. First, the dimension of the vector is significantly reduced from $|\mathcal{M}| \cdot |\mathcal{S}|$ to $|\mathcal{M}|$. Second, the most time-consuming part of the algorithm, namely the cross sum operation, no longer performs on the whole value function but on a smaller vector set.

Acknowledgement

This research work is supported by Hong Kong Research Grants Council Grant: HKUST6152/98E. The authors would like to thank Stephen S. Lee for suggesting the use of the overall improvement as a termination condition for PBI.

References

[1] A. R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes.* PhD thesis, Brown University, 1998.

[2] A. R. Cassandra, M. L. Littman, and N. Zhang. Incremental pruning: A simple, fast, exact algorithm for partially observable Markov decision processes. In *UAI*, 1997.

[3] S. P. M. Choi. *Reinforcement Learning in Nonstationary Environments.* PhD thesis, Hong Kong University of Science and Technology, Department of Computer Science, Hong Kong, China, Jan. 2000.

[4] S. P. M. Choi, D. Y. Yeung, and N. L. Zhang. An environment model for nonstationary reinforcement learning. In *Advances in Neural Information Processing Systems 12*, 1999.

[5] S. P. M. Choi, D. Y. Yeung, and N. L. Zhang. Hidden-Mode Markov decision processes. In *IJCAI 99 Workshop on Neural, Symbolic, and Reinforcement Methods for Sequence Learning*, 1999.

[6] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *NIPS 8*, 1996.

[7] M. Hauskrecht and H. Fraser. Planning medical therapy using partially observable Markov decision processes. In *Proceedings of the Principles of Diagnosis*, pages 182–189, 1998.

[8] S. S. Lee. Planning with partially observable Markov decision processes: Advances in exact solution method. MPhil thesis, Hong Kong University of Science and Technology, Department of Computer Science, Hong Kong, 1999.

[9] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Efficient dynamic-programming updates in partially observable Markov decision processes. Technical Report CS-95-19, Brown University, 1995.

[10] M. L. Puterman. Markov decision processes. In D. P. Heyman and M. J. Sobel, editors, *Handbooks in OR and MS*, volume 2, pages 331–434. Elsevier Science Publishers, 1990.

[11] S. Singh and D. P. Bertsekas. Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *NIPS 9*, 1997.

[12] E. J. Sondik. *The Optimal Control of Partially Observable Markov Processes.* PhD thesis, Stanford University, Stanford, California, USA, 1971.

[13] N. L. Zhang, S. S. Lee, and W. Zhang. A method for speeding up value iteration in partially observable Markov decision processes. In *Proceeding of 15th Conference on Uncertainties in Artificial Intelligence*, 1999.

[14] N. L. Zhang and W. Liu. A model approximation scheme for planning in partially observable stochastic domains. *Journal of Artificial Intelligence Research*, 7:199 – 230, 1997.